AD-A231 389

COPY

AAMRL-TR-90-039

SHAPE-MEMORY ALLOY TACTICAL FEEDBACK ACTUATOR

A. DAVID JOHNSON

TINI ALLOY COMPANY, INC.
1144 65th STREET
OAKLAND, CA 94608

DTIC
ELECTE
JAN 31 1991
S D
D

AUGUST 1990

PHASE I - FINAL REPORT, Air Force SBIR Contract F33-88-C-0541
1 February through 1 August 1989

Approved for public release; distribution is unlimited

ARMSTRONG AEROSPACE MEDICAL RESEARCH LABORATORY
HUMAN SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-6573

91 1 29 019

NOTICES

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Harry G. Armstrong Aerospace Medical Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield VA 22161

Federal Government agencies and their contractors registered with Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
Cameron Station
Alexandria VA 22314

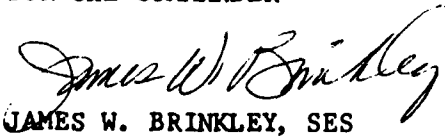
TECHNICAL REVIEW AND APPROVAL

AAMRL-TR-90-039

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



JAMES W. BRINKLEY, SES
Director
Biodynamics and Bioengineering Division
Harry G. Armstrong Aerospace Medical Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1 Sep 89	Final 1 Feb 89 - 1 Aug 89

4. TITLE AND SUBTITLE

Shape-Memory Alloy Tactile Feedback Actuator

5. FUNDING NUMBERS

PE - 65502F
PR - 7231
TA - 38
WU - 04

6. AUTHOR(S)

A. David Johnson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

TINI Alloy Company, INC
1144 65th Street
Oakland CA 94608

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Biodynamics and Bioengineering Division (AAMRL/BBA)
Harry G. Armstrong Aerospace Medical Research Laboratory
Human Systems Division
Air Force Systems Command
Wright-Patterson AFB, OH 45433-6573

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AAMRL-TR-90-039

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This document describes the hardware developed to realize a shape memory alloy (SMA) tactile stimulator. A 5X6 30 element array has been developed to provide tactile stimulation to the human operator of a remotely controlled robotic system. The work verifies that a tactile stimulator can be packaged in finger-pad size arrays and that computer control is feasible. Actuation profiles and design techniques are discussed.

14. SUBJECT TERMS

Shape Memory Alloy Tactile stimulation
Human Sensory feedback

15. NUMBER OF PAGES

35

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

CONTENTS

Accesion For

NTIS CRA&I

DTIC TAB

Unannounced

Justification

By

Distribution /

Availability Codes

Dist | Avail and/or Special

A-1

Overview:

Tactile feedback is expected to be an integral part of future tele-robotic and virtual cockpit instrumentation. In creation of artificial reality environments, the sense of touch has not been exploited as a source of information to the same degree that sight and sound have been, yet it is understood to be fundamentally important to orientation and 'navigation' in the real world. Touch is crucially important to visually deprived individuals who successfully substitute touch for many of the functions usually reserved for vision. It is expected that tactile feedback will prove a strong complement to the other senses in interpretation of an artificial reality.

Basic research in tactile stimulation leading to tactile feedback algorithms is hampered by a lack of suitable hardware devices. It is desirable that lightweight, compact arrays of tactile stimulators be implemented in gloves or other portable form, and that these arrays, which should contain a multiplicity of individually actuated elements, should be computer controlled for sensitive response. The objective of this study is to investigate one approach to creating a tactile feedback device.

The technology employed utilizes heat-actuated wires of titanium-nickel (TiNi) shape-memory alloy (SMA) to move small spring cantilevers so that activated elements rise above a touch-pad surface to be sensed by a finger. Heating of the shape-memory wires is accomplished by electrical current directly through the TiNi wires under control of a digital microcomputer.

Tactile array devices fabricated in Phase I have been demonstrated for the Armstrong Aerospace Medical Research Laboratory (AAMRL) personnel and establish that distinctive patterns presented by this method are readily sensed by the fingertip. This demonstration satisfies the requirements of the Phase I contract. Hardware delivered to AAMRL will be used to evaluate methods of presenting sequences of patterns to subjects. In a second phase of research, this work will be extended to the generation of tactile feedback systems which will be useful for tele-robotics and virtual cockpit simulation.

Background.

This research addresses Solicitation AF88-073: "Force Reflection and Tactile Feedback Technology". The stated requirement was for 'tactile stimulation mechanisms operating in a virtual or telepresence environment that measure and communicate stimulation in three-dimensional space.' It is understood from this solicitation, and from conversations with Capt. Ronald Julian and Capt. George Williams, that the ideal tactile stimulator should be multiple-element, compact enough to fit within an instrumented glove, and should respond fast enough and with enough force to create an "artificial reality" of touch. Fine wire of nickel-titanium SMA is capable of providing the necessary force and motion in a very compact configuration, and TiNi has suitable physical characteristics for electrical actuation.[Ref. 1] Work accomplished in Phase I is a step in realization of the goal of employing shape-memory technology to create a compact, light-weight tactile stimulator array.

Limited literature on tactile stimulation leads us to conclude that basic research in tactual perception will be improved with access to computer-driven dynamic tactile stimulators.[Refs. 2-6] Static displays of literal information in Braille form, in which the finger moves across raised dots, stimulate the tactile senses in ways which are quite different from devices intended to convey artificial reality. Psychophysical mechanisms for extracting information on texture, hardness, and other surface characteristics clearly depend on time-dependent signals. The Optacon[Ref.3] uses vibration to compensate for diminution of tactual sense to constant stimuli. It produces large-amplitude time-varying signals. Such devices are not ideal as a means of transmitting textual information and are probably incapable of conveying information such as surface texture. A useful tool for study of the basis of tactual perception seems to require actuators capable of a steady-state signal upon which a time-varying vibratory signal may be superimposed.

Vastly more subtle effects will need to be understood if a user of artificial reality is to be able to distinguish by touch between a glass surface and a metallic surface. Creation of artificial reality will not be complete until this type of information is available. How this will be accomplished is difficult to speculate. The research which we have begun is, necessarily restricted to perception of displacement as a function of time.

Hardware.

The requirement for a lightweight tactile actuator with a low profile has been met by an array of 30 cantilevers machined from a sheet of beryllium copper(BeCu) alloy. This array was fabricated by chemical milling of soft BeCu in a pattern shown in Figures 1A and 1B. The free end of each beam was subsequently bent at a right angle. The result is a rectangular pattern 5 by 6 elements separated by approximately 3.0 mm. After bending, the array was heat treated to increase elasticity of the BeCu alloy.

Individual beams, approximately .75 mm wide and 10 mm long, are deflected upward elastically by contraction of a shape memory wire. The actuator configuration resembles a tendon which extends a phalange in a human finger, as seen in Figure 2. To each beam is connected a wire of TiNi shape-memory alloy (SMA) 15 mm long and .1 mm in diameter. TiNi SMA has the property that it elongates easily at room temperature but contracts strongly when heated by a pulse of electrical current. The wire is connected so that as the SMA wire contracts it bends the cantilever upward. An upwardly projecting tip on the beam, formed by bending the tip of the cantilever beam at a right angle, extends above the touch surface about .5 mm when the TiNi wire is in its contracted state. When electrical current is turned off, the cantilever spring elongates the SMA and the tactile element retracts.

The three principal hardware challenges to be met were; manufacture of the array; mechanical attachment of the TiNi wire to the BeCu plate; and making electrical contact at both ends of the TiNi wire.

An experimental array was formed by sawing with a slitting saw. This method did not maintain proper dimensions, and produced only one part for several hours of labor. Laser cutting was done, with the result that very poor quality edges were produced. Chemical etching by photoresist pattern proved to be a satisfactory method which produced high quality results, at a relatively low cost per part.

At the distal end of each beam, three small holes are spaced about 1 mm apart. The TiNi wire is woven through these holes, which serve to anchor it mechanically and to provide reliable electrical contact with the BeCu. In order to stabilize this connection mechanically and to reduce the probability of open electrical circuits, conductive epoxy was applied at each of the entry holes.

At the other end of the cantilever beam, the TiNi wire is attached to a Delrin bridge which is slotted to give the wires proper spacing. A stainless steel tube, 0.7 mm diameter and 3 mm long, is used to make electrical and mechanical connection. The TiNi wire is swaged into one end of the tube and at the other end a silver wire is swaged to form a solder connection to the electrical cable.

Actuator design.

Using the elastic properties of the beryllium copper substrate and the thermal contraction of the TiNi wire, a design was achieved by iterated computer calculation.

Each tactile element consists of a BeCu beam to which is attached an SMA wire as shown in Figures 1 and 3. Electric current through the SMA wire causes it to heat and contract about 3 percent with a stress of up to 210 kPa. An SMA wire 0.076 mm in diameter and 15.24 mm long contracts approximately 0.508 mm, exerting a force of up to 70 grams when heated with a current of 0.2 amperes.

Deflection of a beam by a load force at one end is described by the relationship

$$y = F_S * x^3 / (3 * E * I) \qquad \text{[EQ. 1]}$$

where y is deflection
F_S is spring force
x is beam length
E is elastic constant of beam
I is moment of inertia about center of beam.

For a beam of rectangular cross-section,

$$I = b * h^3 / 12 \qquad \text{[EQ. 2]}$$

where b is beam width
h is beam thickness.

Slope of the beam at the deflected end is

$$dy/dx = 3\ k * x^3 = \tan(\text{theta}) \qquad \text{[EQ. 3]}$$

where $k = F_S / 3*E*I$
and theta = dip angle of deflected end of beam.

From which

$$F_S = E * b * h^3 * y / 4 * x^3 \qquad \text{[EQ. 4]}$$

If the SMA wire pulls horizontally with force F_N, the force perpendicular to the beam (F_B) is

$$F_B = F_N \sin(\text{theta}) = 3 * F_N * y/\text{sqrt}(x^2 + 9\ y^2) \qquad \text{[EQ. 5]}$$

The force available for tactile 'feel' (F_T) is the difference

$$F_T = F_B - F_S \qquad \text{[EQ. 6]}$$

The result of calculation using a typical set of values for beam dimensions is presented in Table I.

Hardware Function.

The electronic control hardware provides a constant-current source for each of the 30 actuators. The instantaneous magnitude of the current depends on the values of the sense and control resistors in the circuit depicted in Figure 4. The values given produce a current through the SMA wire of about 0.18 amp.

The actuating signal for an actuator consists of a starting pulse followed by a train of pulse-width-modulated (PWM) pulses. Note that the magnitude of the current is the same for the starting pulse as for each of the modulated pulses. PWM provided by the on-board microprocessor through a latched set of current-regulated transistors determines an average current sufficient to maintain the SMA wire in its contracted configuration without overheating it. Length of the starting pulse, duty cycle of the PWM signal, and frequency of PWM are under software control.

Rise-time measurements.

Preliminary data on actuation time as a function of pulse current density was made using a data-acquisition system connected to an IBM-PCXT computer. This system is depicted schematically in Figure 5.

One 'finger' of the tactile pad array was attached to an LVDT to measure travel as the finger was lifted by contraction of the SMA element. A computer-controlled pulse of current was used to actuate the element, and its response time vs. displacement were recorded by a data acquisition system. From these experiments a set of response versus time curves were plotted, each at a specified current pulse height. This shows that an actuation time of .1 sec is feasible using a current pulse of .24 ampere for .1 second duration. Recovery time is approximately one second. See Figures 6A and 6B.

Electronics.

The drive circuit for the individual tactile elements is shown schematically in Figure 4. This circuit uses feedback from a 5.1 ohm sense resistor to supply constant current to the TiNi actuator. A 2N2222 transistor controls the gate voltage to an IRF640 power MOSFET permitting current (approximately .18 ampere with the circuit parameters used) from the +5 volt power supply to flow through the SMA, FET, and sense resistor, to ground. Voltage to the collector of the 2N2222 transistor is controlled by the state of the corresponding latch in the 5818 serial-to-parallel converter, which in turn stores the pattern sent via serial port from the Motorola MC68HC11 microprocessor. Information on components used in this circuit are included in Appendix B.

The microprocessor board from New Micros has built-in FORTH language, and all of the logic for controlling the tactile elements has been programmed in this language and stored in the on-board eprom. Communication with the host computer is via an RS232 port on the microprocessor.

A command to set a particular pattern into the tactile pad consists of a string of characters sent from the host computer, placed on the stack of the microprocessor, and interpreted by the FORTH program. Patterns to be transmitted are selected by the operator from a software menu. Translation of bit patterns to pin numbers on the connector cable, and to the corresponding individual tactile elements, are defined in Table II and Figure 7.

The microprocessor board is described in documentation from the manufacturer. See Appendix C.

Software.

Host computer programming for the tactile feedback device is in MicroSoft Quick-Basic 4 language, intended to be run on an IBM PCXT or compatible. The program is invoked by executing TAC.EXE, provided on a magnetic disk. A menu of options is provided. These options permit the user to:

- Create new patterns to be displayed via the tactile pad;
- "Teach" these patterns so that they may be repeated;
- Store patterns for retrieval at a later time;
- Send individual patterns to the microprocessor;
- Step from one pattern to another by means of right and left arrow keys;
- Cycle through a selected set of patterns repeatedly, with timing specified by the user.

Two example programs are listed in Appendix D. Source code and object programs are provided on the floppy disk. The first is TAC.BAS (the source code for TAC.EXE), which is the principal operating program for the tactile device. The second is SIMPLE.BAS, which is a simplified program intended to be used as a tutorial by anyone who wishes to create his own working version of the tactile communications package.

Appendix E is a listing of FORTH code written for the dedicated microprocessor which receives ASCII strings from the host computer, interprets them and drives the tactile array.

A detailed step-by-step set of operating instructions is provided in Appendix A.

Summary and conclusions.

A 30-element tactile finger pad has been constructed and demonstrated. Minimum actuation time is approximately 0.1 second for an individual element. Force exerted against the finger tip by an individual stimulator is of the order of 20 grams. Relaxation time for the device with actuators in still air is about one second. The device is unlikely to cause electrical shock as the only power delivered through the tactile actuator array is low amperage direct current with a voltage potential of five volts.

Electronics hardware and software are provided to control individual tactile elements. Control of patterns of tactile elements is through a host computer by RS232 communications. Sample patterns and examples of control sequences, programmed in Quick-Basic 4.0, are provided on floppy disk.

The prototype device has been demonstrated for Air Force personnel and for a selected audience. The reaction has been generally favorable. Nearly everyone can detect a sense of motion when a succession of lines of the tactile stimulator array are activated in succession.

A finding of possible significance is that the pulse-width modulation of the heating current in the TiNi actuators produces a vibration which is easily felt by touch. At a frequency of about 20 Hz, this vibration is sufficient to overcome the tendency of the sense of touch to become accommodated to a constant signal, so that the tactile sense is enhanced. In future studies, this feature should be studied to ascertain the optimum frequency (related to the cooling rate of the TiNi wire) and magnitude of vibration to enhance the sense of touch.

In the prototype tactile stimulator array there is no provision for adjustment of the height of the individual actuators. It is difficult, because of the small size and the fragility of the components, to achieve and maintain uniform tension on the TiNi actuator wires. As a result, performance of the prototype device is uneven. In succeeding generations of this device it is hoped that a simple method of adjusting the SMA wires will evolve, or else that a method of stopping the rise of the pins will be worked out so that uniform, consistent pin actuation will result.

The 30-element array constructed in the course of this research is near the limit of what can be constructed manually because of the difficulty and expense of assembling small components. It may be desirable to make even closer-packed arrays in order to study the limits of tactile perception of minute details. In this case it is recommended that alternative methods of fabrication be applied. Microfabrication techniques including sputter deposition of TiNi in the form of thin foils offers an opportunity for fabrication of much smaller elements than can be achieved by construction using individual components.[Ref.8]

A limitation of the Phase I device is the slow recovery due to the cooling rate of wire in ambient air. Prior experiments have shown that it is feasible to cool with moving air to increase cooling speed, and that by immersion in fluid complete cycle rates up to 10 hertz are possible. Cooling rates are dependent on the surface-to-volume ratio and on the cooling medium.

In connection with increasing the speed of the device, it should be noted that it is not really known what data rates are necessary to transmit useful data via the sense of touch. In the Phase I device, the onset of stimulus is rapid and sustained stimulus is aided by vibration from the pulse-width-modulated electrical signal. Probably the cessation of the vibration which accompanies the cooling of the SMA wire helps in perception of signal termination. It is hoped that the devices built in the course of this research will be a useful tool for evaluation of questions related to the sense of touch.

The immediate application envisioned for tactile feedback is in tele-operator and virtual cockpit studies. In a Phase II, TiNi Alloy Company will propose to fabricate tactile stimulator devices and conduct research into basic questions concerning the best means of presenting tactile information for the above applications. The objective of this research is a tactile module which may be worn in a glove to provide tactile feedback in conjunction with other sensory input.

It has been suggested [Ref.9] that a programmable multiple-element tactile stimulator may be useful in university research and for clinical evaluation of tactile response in people suffering the effects of stroke or other sensory loss. The opportunities for this and other possible markets will be investigated.

Acknowledgements:

Aram Soghikian, TQ Automation Company, was responsible for design of electronics and software.

John Brabyn, an employee of Smith-Kettlewell Eye Research Institute in San Francisco, invented the bending-beam tactile actuator. He aided in layout of the grid of bending beams to form a rectangular array of tactile elements as well as preliminary calculations of the design parameters.

This research was supported by Air Force SBIR contract number F33-88-C-0541, Sonia K. Carlton contracting officer, PMRSB, Wright-Patterson AFB, OH, 45433.

References.

[1]. Hiroyasu Funakubo, "Shape Memory Alloys", Precision Machinery and Robotics Vol. 1, Translated by J.B. Kennedy, Gordon and Breach, 1986.

[2]. Ronald G. Julian and Timothy R. Anderson, "Robotic Telepresence: Applications of human controlled robots in Air Force maintenance", Aerospace Simulation 1988. ISBN 0-911801-28-6.

[3]. J.C. Bliss, H.D. Crane, S.W. Link, and J.T. Townsend, "Tactile Perception of Sequentially Presented Spatial Patterns," Perception and Psychophysics, Vol. 1, pp. 125-130, 1966.

[4]. Cecilia Nguyen, Masters Thesis, University of California at Berkeley under Prof. Lawrence Stark.

[5]. Zenon Kuc, "A Vibrotactile Communication System: Tactual Display Design and Attainable Data Rates", Center for Integrated Systems, Stanford University, Stanford, CA 94305, 1989. Zenon Kuc is writing a Ph.D dissertation on related research.

[6]. Kenneth J. Kokjer, "The Information Capacity of the Human Fingertip", IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-17, No. 1, January/February 1987.

[7]. Kent, Mechanical Engineers' Handbook, 12th edition, John Wiley & Sons, pp. 8-12.

[8]. A.D. Johnson and J.D. Busch, "Digital Storage Device using Thin Film Shape Memory Alloy", Phase I final report, NASA SBIR Contract NAS2-12797, TiNi Alloy Company, Oakland, CA. 1989.

[9]. Prof. Arnold Leiman, Psychology Department, University of California at Berkeley. Private communication.

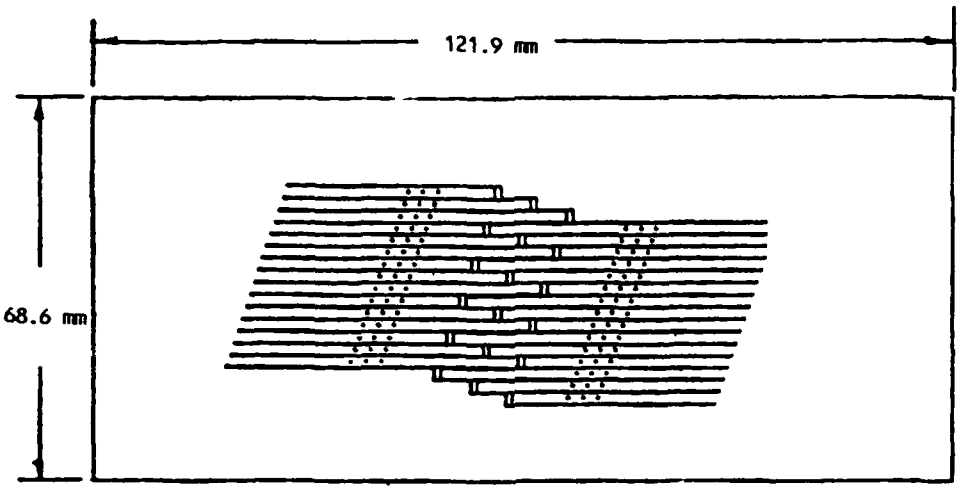


Figure 1 A. Pattern Cut in BeCu to form Tactile Stimulator Elements.

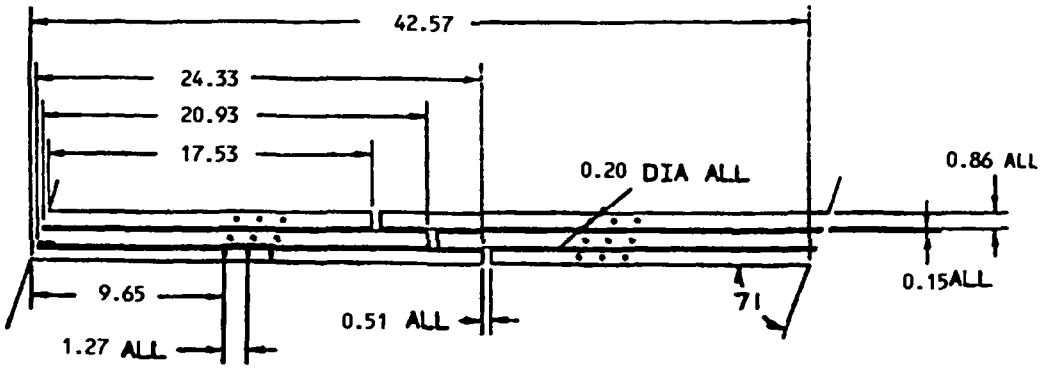


Figure 1B. Detailed element dimensions(all dimensions on millimeters)

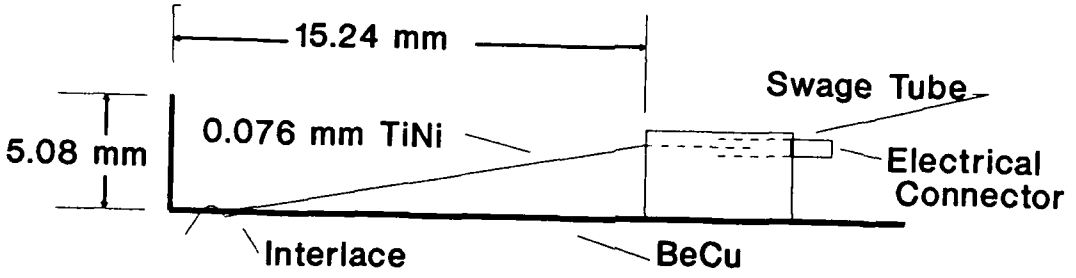


Figure 2. Tactile Stimulator Element

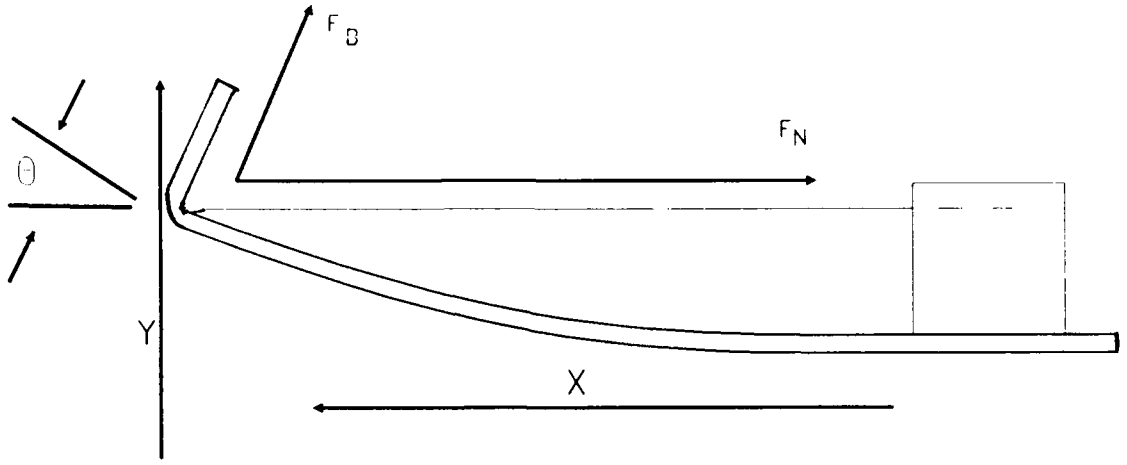


Figure 3. Tactile Stimulator in Raised Position

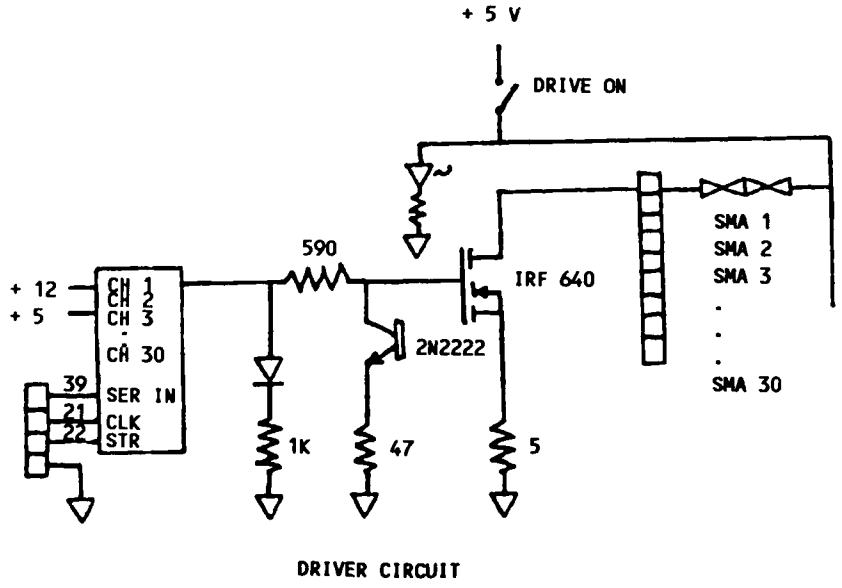


Figure 4. Schematic of Constant Current Supply for Tactile Stimulator Element

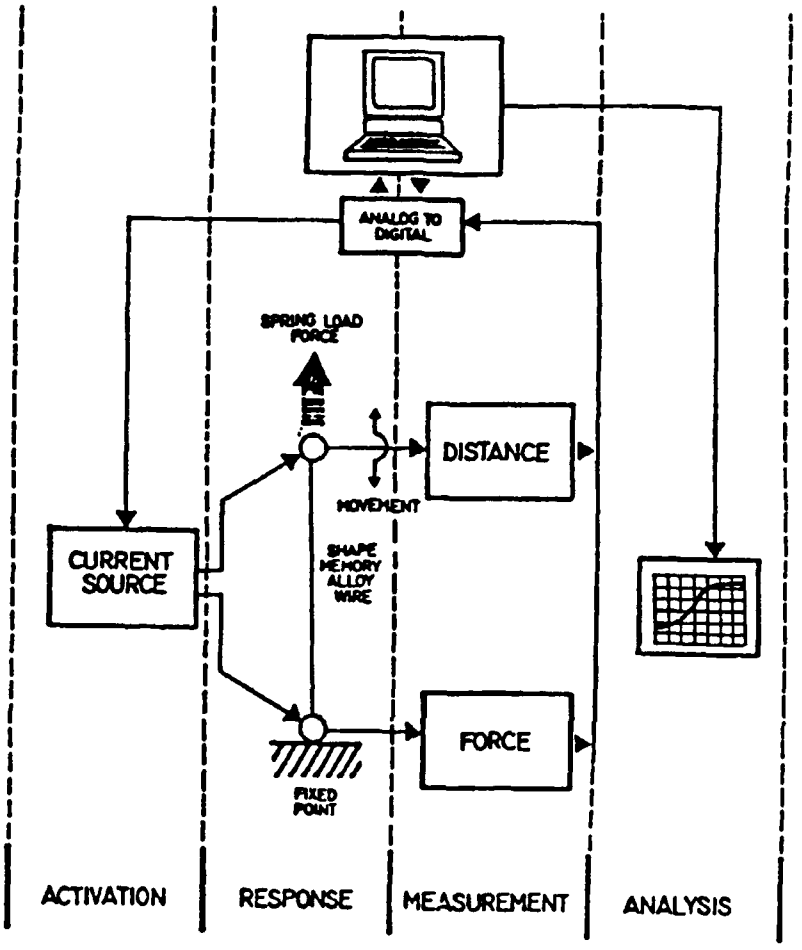


Figure 5. Data Acquisition System Used to Measure Time Response of Tactile Elements.

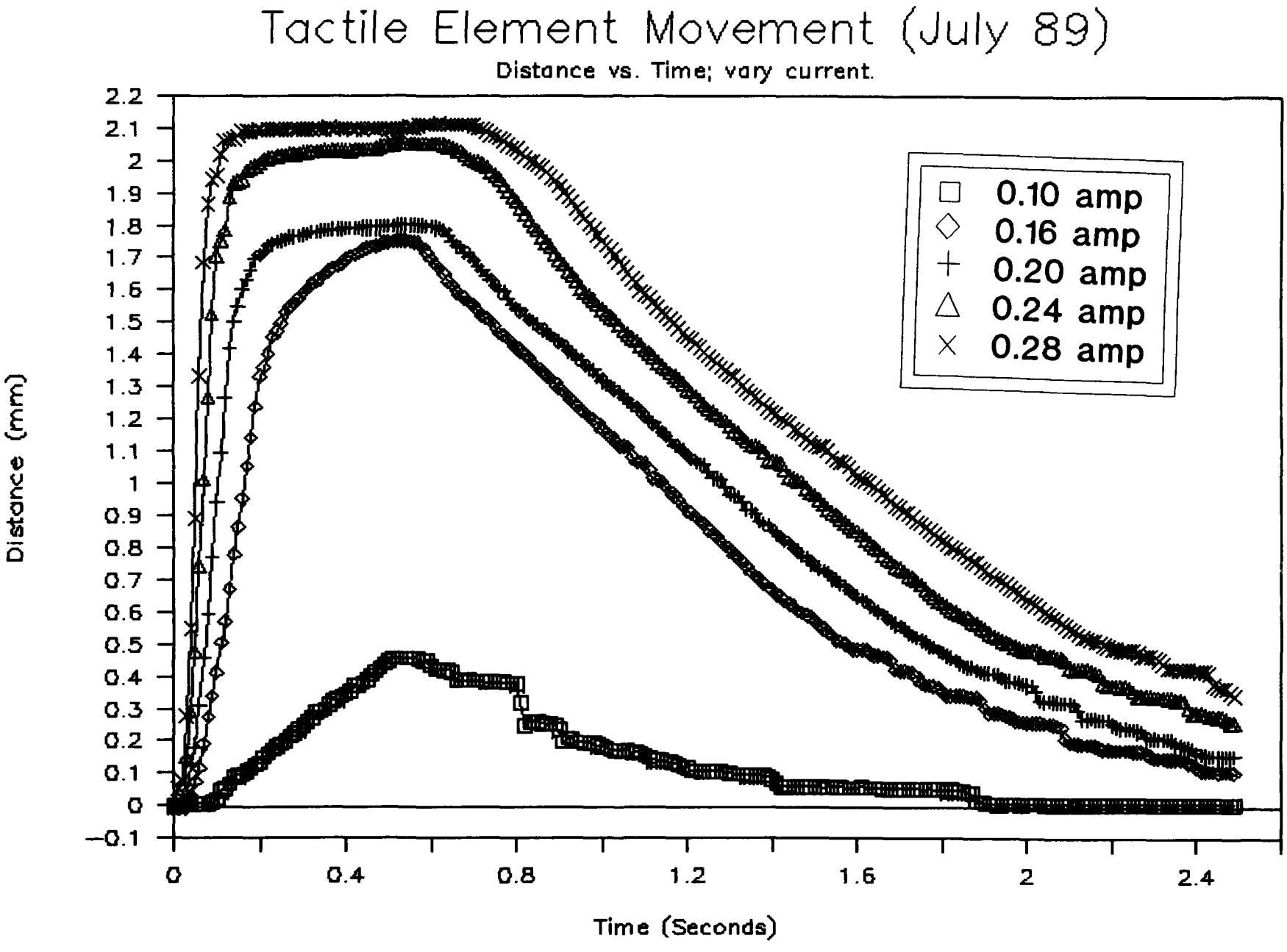


Figure 6A. Time Response of TiNi Actuator Wire to Pulses of Electrical Current

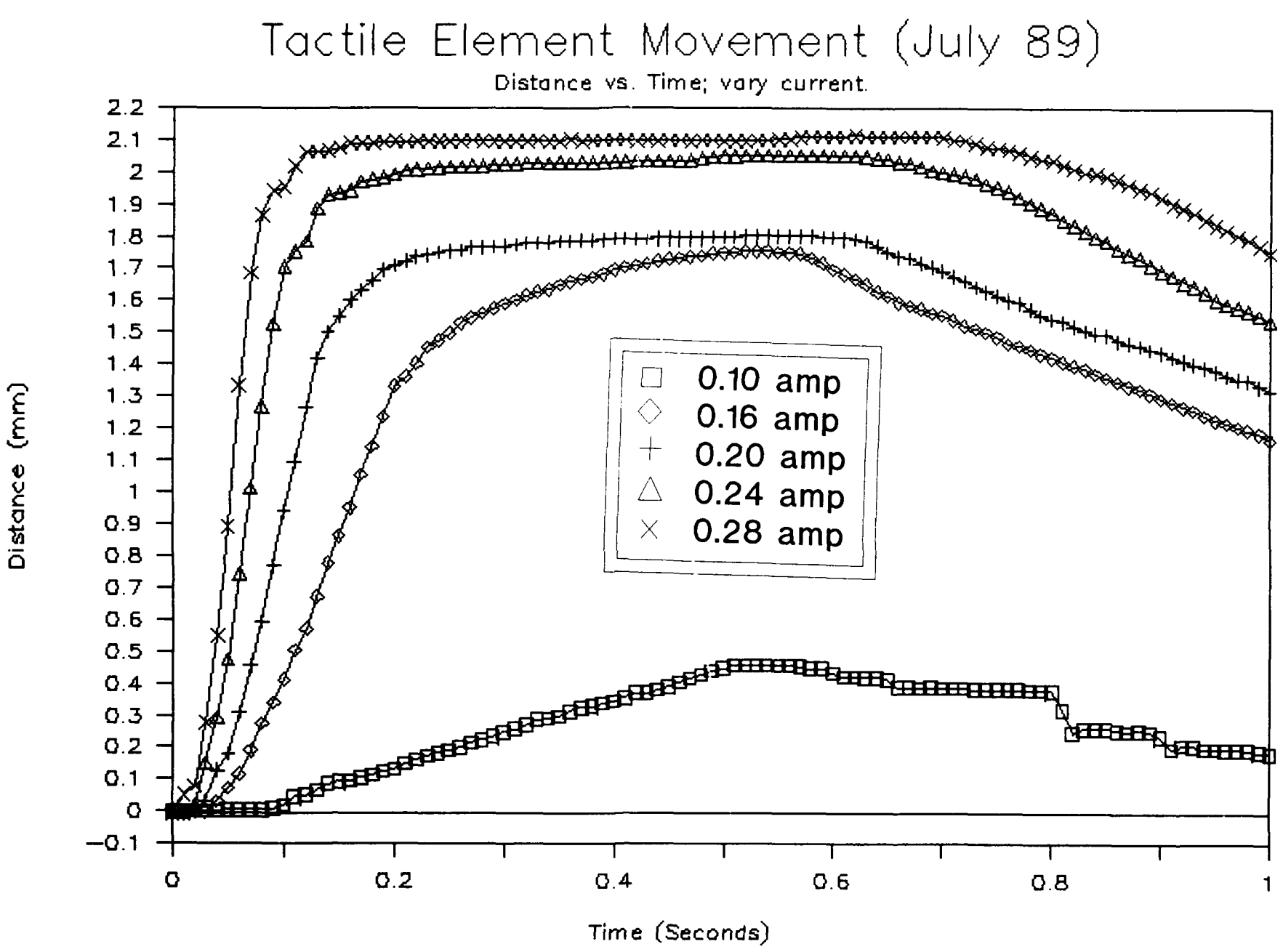


Figure 6B. Expanded Scale of Figure 6A.

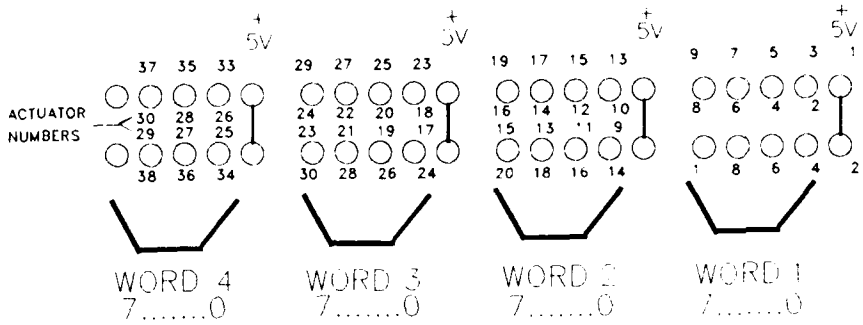


FIGURE 7A: WIRING TABLE: CONNECTOR(TOP VIEW) TO TACTILE ACTUATORS

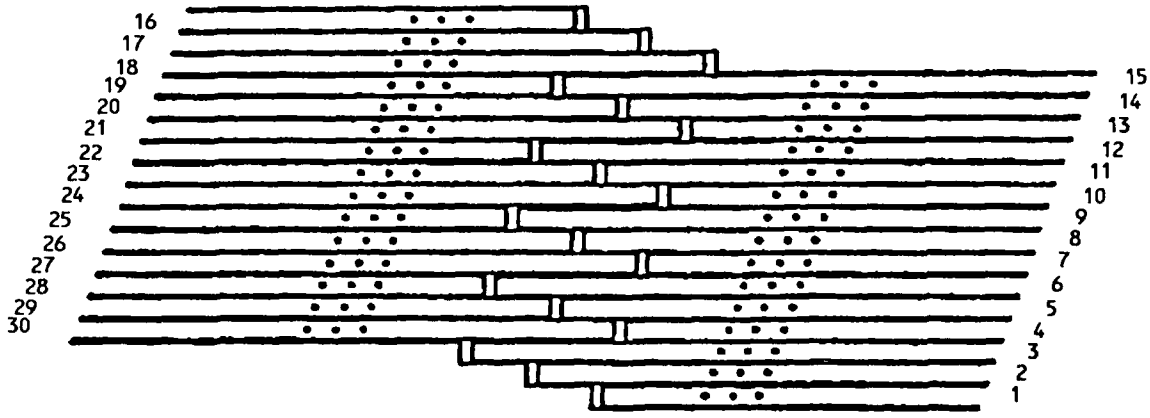


Figure 7B. Correspondence of Tactile Stimulator Position with Electrical Cable connection and with Bit Position in Control Word.

TABLE I

BEAM CALCULATION

BEAM LENGTH X	0.380	INCHES	input
VERTICAL DISPLACEMENT Y	0.075	INCHES	input
BEAM WIDTH B	0.032	INCHES	input
NITINOL FORCE FN	0.085	POUNDS	input
YOUNG'S MODULUS E	18.5	MPSI	input
Wire diameter	1.9	Mils	r^2=FN/Pi*Stress
CONSTANT K	1.367		y/x^3
ANGLE THETA	0.535	RADIANS	Atan 3kx^2
DEFLECTING FORCE FB	0.043	POUNDS	FN sin theta
BEAM THICKNESS EQUATION:			
TOP LINE	0.173		
BOTTOM LINE	2.427E+06		
FORMULA	7.136E-08		
BEAM THICKNESS H	0.004	INCHES	
APPROX. NITINOL CONTRACTION	1.9	%	

REVISED PROGRAM - August 1989 ADJ

Beam Length	l		0.380 inches
Beam Width	b		0.032 inches
Beam Thickness	h		0.006 inches
Beam Displacement	y		0.075 inches
Beam Material Elasticity	E	BeCu	18.5 Mpsi
Nitinol Wire Diameter	D		0.003 inches

From KENT pg.8-12, P=3*E*I*y/l^3, I=b*h^3/12
Hence: Spring force perp. spring end = E*b*h^3*y/4*x^3

Spring Force Working Against Nitinol	lbs	0.044	19.861
Nitinol Force Capability at 30,000 psi	lbs	0.212	96.390
Displacement Force due to TiNi	lbs	0.108	49.110
Resulting Tactile Force	lbs	0.064	29.249
Contraction Percent		1.9	

TABLE II

CORRESPONDENCE BETWEEN BIT PATTERNS, CONNECTOR PINS, AND TACTILE ACTUATOR LOCATIONS

WORD	BIT	CONNECTOR PIN	ACTUATOR NUMBER	ROW	COLUMN
1	0	3	1	5	3
	1	4	2	5	2
	2	5	3	5	1
	3	6	4	4	3
	4	7	5	4	2
	5	8	6	4	1
	6	9	7	3	3
	7	10	8	3	2
2	0	13	9	3	1
	1	14	10	2	3
	2	15	11	2	2
	3	16	12	2	1
	4	17	13	1	3
	5	18	14	1	2
	6	19	15	1	1
	7	20	16	1	4
3	0	23	17	1	5
	1	24	18	1	6
	2	25	19	2	4
	3	26	20	2	5
	4	27	21	2	6
	5	28	22	3	4
	6	29	23	3	5
	7	30	24	3	6
4	0	33	25	4	4
	1	34	26	4	5
	2	35	27	4	6
	3	36	28	5	4
	4	37	29	5	5
	5	38	30	5	6
	6	NC			
	7	NC			

TACTILE DEVICE OPERATING INSTRUCTIONS

This document is intended tc introduce the TiNi Alloy Company tactile stimulation array device to the user without prior experience. Problems with the device or comments concerning this document should be addressed to:

Dr. A. David Johnson
TiNi Alloy Company, Inc.
1144 65th Street, Oakland, CA 94608.
Telephone (415) 658-3172 FAX (415) 658-3764.

SET-UP:

Connect power supply to chassis using the 9-pin connector. Turn "DRIVE ON" switch to left (off) position. In this position, no current will flow in the tactile actuator, which will protect it in case the circuit, for any reason, does not come up in normal operating mode.

Connect RS232 host computer port to chassis using 9-pin connector. Port 1 (COM1 9600 Baud, 8bits,1 stop bit, no parity.)

> CAUTION. Observe the markings. Do not connect power to RS232 or vice versa.

Connect host computer and piggy-back power supply to 110 volt AC.

Power on Host PC. Power on Power supply.

Access the directory where the TAC.EXE file resides. For example, A:TAC.EXE if using the floppy disk version.

Load the program, with optional timing parameter and port specification. For example, type

TAC .4 COM1 (enter)

to get .4 sec heating time for bringing the tactile elements to the UP position. For the specified time, full current will be applied to the actuators for the tactile elements, after which pulse-width modulation will be used to hold them in position.

> CAUTION: too long a time will damage the shape-memory wires which do the lifting.

[It is recommended that a batch file be placed in the root directory to make access simple. Something like

A:
TAC .4 COM1

appropriately amended to point to the directory where the system is loaded on hard disk: the essential files are TAC.EXE, KEYBITS.ARY, and one or more *.PAT files such as BITS.PAT.]

As the program loads it will display a brief text, then a menu:

```
"  TACTILE ARRAY DRIVER COMMANDS:"
"   Left arrow = Back step"
"  Right arrow = Foreward step"
"       Ctrl-T = Teach pattern"
"       Ctrl-L = Load pattern file"
"       Ctrl-S = Save pattern file"
"       Ctrl-A = Automatic cycling"
"       Ctrl-N = New keyboard layout (CAUTION)"
" F1 or Ctrl-H = Help screen"
"    Other keys = Toggle tactile elements on and off"
"           Esc = Quit"
```

followed by a line representing the 30 bits to be sent to the tactile pad:

```
     00000000 00000000 00000000 00000000
      Char.1   Char.2   Char.3   Char.4
```

Bits 0 through 7 of Char.1 are toggled by the numbers 1 through 8.
Bits 0 through 7 of Char.2 are toggled by letters Q through I.
Bits 0 through 7 of Char.3 are toggled by letters A through K.
Bits 0 through 5 of Char.4 are toggled by letters Z through N.
(The last two bits of Char.4 are unused.)

That is, if you press A the first bit in Char.2 will become 1 and the corresponding light in the control circuit will be turned on. Pressing the A key again will turn the light off.

The correspondence between bits in the four Chars and elements in the tactile pad is defined in Table I.

An important point: the LED (and the tactile element, if the DRIVE ON switch is set to ON) are first given a pulse of full current lasting a fraction of a second to actuate it by heating the TiNi wire. After this initial pulse, a train of shorter pulses is provided to keep the tactile element in its UP position. This pulse-width modulation, which makes the LEDs flicker, is necessary to protect the TiNi wires from becoming overheated. If the PWM is not present, the DRIVE ON switch should be turned left (OFF) as quickly as possible, and maintenance should be alerted.

You may now turn on the DRIVE ON switch, so that the tactile elements corresponding to bits turned to "1" as described above are actuated.

It is wise to keep the DRIVE ON switch in the un-activated position when you are not actively using the display, as either circuit or software failure can leave power on a wire in such a way that it is damaged or destroyed.

You may create a set of patterns. To do so, press CTRL-T ("teach") to advance to a next pattern. When you have created a set of patterns, you may preserve this set by pressing CTRL-S ("Save"). The program will ask you te specify a file name of up to 8 characters. The subtitle .PAT will be automatically appended.

You may retrieve a set of patterns which have been previously saved. Press CTRL-L ("Load"), and you will be prompted by a list of the available files. Choose one and press ENTER and the file will be loaded and the contents displayed.

To move from one pattern to another within a set, use the right-arrow key to advance, left-arrow to back up from one pattern to another.

To cycle automatically through a set of patterns, press CTRL-A ("Auto"). You will be prompted to specify a length of time for which each pattern will be displayed before proceeding to the next pattern. Pressing any key exits from "Auto" mode.

If you forget the menu, you can get it back by pressing CTRL-H ("help") or F1 key.

If the above choice of keys seems inapproriate, you may define an alternate scheme using the CTRL-N ("new keyboard") feature. If you do so, you must keep your own documentation for this new layout, which will be stored in a file named KEYBITS.ARY which is loaded automatically when the TAC program is brought up. In this case it is recommended that you keep a backup file copy of KEYBITS.ARY so that you may return to it easily.

The following prompt appears at the beginning of the program to aid in performing these functions:

"This program allows you to teach patterns on the array and play them back using the arrow keys. You toggle elements on and off by pressing the keys corresponding to each element, as defined by the lookup table saved on disk as KEYBITS.ARY. To redefine the keyboard layout, press Ctrl-N. Then enter 30 keystrokes corresponding to element channel numbers 0 - 29 in ascending order. Doing this will overwrite KEYBITS.ARY, so save it under another name if you want to keep it."

EXITING THE PROGRAM:

Normal exit procedure is to press 'ESC' key. This turns off power to all the tactile elements driven by the microprocessor, so that a graceful power-down is accomplished.

APPENDIX B

The Siliconix IRF 640 Power MOSFET amplifiers have uses as switching regulators, converters, choppers, and audio amplifiers. Complete specifications are available from:

MOS Power Databook(1985) pages 1-47 and 1-48

Siliconix Inc
17821 E. 17th St
Suite 240
Tustin, CA 92680

The Sprague UCN-5818AF converter specifications are available from:

Integrated Circuits Databook(WR-508)
pages 5-52 through 5-54

Sprague Electric Company
Semiconductor Group
115 North Cutoff
Worcester, MA 01606

Additional information on the HC MOS MC68HC11A8 is available from:

Motorola Databook DL139
Microprocessor, Microcontroller and Peripheral Data
pages 3-1518 through 3-1539

Motorola Inc
Microprocessor Products Group
Microcontroller Division
Austin, TX 78735

Additional information fo the '100 Squared' system is available from

New Micros Inc
1601 Chalk Hill Rd
Dallas Tx 75212

```
                '*****************************************
                '*        AF TACTILE DEVICE DRIVER       *
                '*          For Tini Alloy Co.           *
                '*         TQ Automation, 6/89           *
                '*****************************************
'This version has a timing loop and some hand shaking removed from SEND

DECLARE SUB SEND (OutStr$)     'Subroutine that sends data to the 68HC11
MaxPatterns = 500              '500 possible stored patterns
DIM Pattern&(MaxPatterns)      'Pattern storage array
DIM Layout&(256)               'Keyboard layout lookup table
DIM BitMask&(32)               'Only 30 bits used because BASIC barfs on 2^31 AND <anything>
GOTO START                     'Skip HELPSCREEN

HELPSCREEN:  '*********  HIT CTRL-H OR F1 TO GET THIS SCREEN  *************
PRINT
PRINT "  This program allows you to teach patterns on the array and play them"
PRINT "back using the arrow keys.  You toggle elements on and off by pressing"
PRINT "the keys corresponding to each element, as defined by the lookup table"
PRINT "saved on disk as KEYBITS.ARY.  To redefine the keyboard layout, press"
PRINT "Ctrl-N.  Then enter 30 keystrokes corresponding to element channel numbers"
PRINT "0 - 29 in ascending order.  Doing this will overwrite KEYBITS.ARY, so"
PRINT "save it under another name if you want to keep it."


PRINT
PRINT "  ****  TACTILE ARRAY DRIVER COMMANDS  ****"
PRINT "         Left arrow = Back step"
PRINT "        Right arrow = Foreward step"
PRINT "             Ctrl-T = Teach pattern"
PRINT "             Ctrl-L = Load pattern file"
PRINT "             Ctrl-S = Save pattern file"
PRINT "             Ctrl-A = Automatic cycling"
PRINT "             Ctrl-N = New keyboard layout (CAUTION)"
PRINT "  F1 or ? or Ctrl-H = Help screen"
PRINT "         Other keys = Toggle tactile elements on and off"
PRINT "                Esc = Quit"

RETURN


START:
CLS : PRINT "     ****  TACTILE DEVICE CONTOLLER  ****"
GOSUB HELPSCREEN

SETUP:     '********  DEFINE VARIABLES  ****************
   Cntr = 1                          'Pattern number counter
   Npatterns = Cntr                  'Total number of pattens taught
   Current& = 0                      'Current actual bit pattern befor teaching
   HotTime! = .3                     'Hot pulse time [sec] added to program latency
   IF COMMAND$ = "" THEN
      PRINT "NOTE:  You need to select the serial port and specify hot cycle time"
      PRINT "Syntax: TAC <hot cycle time, ms> <COM1 or COM2>"
      PRINT "Example: TAC .4 COM1";
      END
   END IF
   IF VAL(COMMAND$) > 0 THEN HotTime! = VAL(COMMAND$)  'Command line entry of hot cycle time
   IF INSTR(UCASE$(COMMAND$), "COM2") THEN
      Port$ = "COM2:"
   ELSE
      Port$ = "COM1:"
   END IF
   IF HotTime! < .01 THEN HotTime! = .01   'Force Pause to be within limits
   IF HotTime! > 3 THEN HotTime! = 3
   Esc$ = CHR$(27)                   'Escape key, quit
   Rarrow$ = CHR$(0) + CHR$(77)      'Right arrow key, forward step
   Larrow$ = CHR$(0) + CHR$(75)      'Left arrow key, back step
   Teach$ = CHR$(20)                 'Ctrl-T, teach pattern
   New$ = CHR$(14)                   'Ctrl-N, New keyboard layout
   Help$ = CHR$(8)                   'Ctrl-H, help screen
   Load$ = CHR$(12)                  'Ctrl-L, load file
```

```
   Save$ = CHR$(19)                'Ctrl-S, Save file
   F1$ = CHR$(0) + CHR$(59)        'F1 key, also help screen
   Auto$ = CHR$(1)                 'Ctrl-A, auto cycle through patterns

   PRINT "...Loading keyboard layout"; : LOCATE , 1
   OPEN "KEYBITS.ARY" FOR INPUT AS #1      'Read in keyboard layout array
      FOR Char = 0 TO 255
         INPUT #1, Layout&(Char)       'Lookup table Character <--> Bit
      NEXT Char
   CLOSE #1
   PRINT SPACE$(40); : LOCATE , 1

   FOR Bit = 0 TO 29
      BitMask&(Bit) = 2& ^ Bit
   NEXT Bit


'        *****   DRIVER BOARD INITIALIZATION  *****

   PRINT "...Initializing the driver board"; : LOCATE , 1
   OPEN Port$ + " 9600,N,8,1,CD0,CS0,DS0,OP1000,RS, " FOR RANDOM AS #2
   SEND " "                        'Clear the terminal input buffer
   SEND "58off"
   SEND "COLD"                     'Do a cold reset on the NMI board
   SEND "HEX"                      'Switch to hexadecimal number base
   SEND "8000 Period !"            'Set cycle period (lower = higher frequency)
   SEND "6000 Ontime !"            'Set duty cycle (lower = shorter duty cycle)
                                   '    ... duty cycle = Ontime / Period
   SEND "58on"                     'Turn on the PWM timer
   PRINT SPACE$(40); : LOCATE , 1


   PRINT : PRINT : GOSUB DRIVE    'Clear all elements, write screen

MAINLOOP:    '*******  THIS IS THE PROGRAM  *********
   DO: Key$ = UCASE$(INKEY$)       'Hang here till a key is pressed
      IF TIMER > TimeOut! AND hot THEN SEND "x": hot = 0    'Wait hot cycle, then go to PWM sustain
cycle
   LOOP WHILE Key$ = ""
   WHILE INKEY$ <> "": WEND          'Dump keyboard buffer

   SELECT CASE Key$       'Take action based on the keystroke received
      CASE Esc$              'Escape ends program
         Current& = 0
         Cntr = 0
         GOSUB DRIVE
         SEND "58off HEX"   'Stop PWM timer"
         CLOSE #2
         PRINT : PRINT "   Thank you ": PRINT
         END
      CASE Larrow$           'Left arrow back steps through pattern counter
         IF Cntr > 1 THEN
            Cntr = Cntr - 1
            Current& = Pattern&(Cntr)
            GOSUB DRIVE
         END IF
      CASE Rarrow$           'Right arrow forward steps through patterns
         IF Cntr < Npatterns - 1 THEN  'Scrolling only goes up to last tought pattern
            Cntr = Cntr + 1
            Current& = Pattern&(Cntr)
            GOSUB DRIVE
         END IF
      CASE Auto$             'Automatic timed display
         INPUT "Time to pause between patterns"; Pause!
         Pause! = INT(Pause! * 100) / 100  'Truncate to 2 decimal places
         IF Pause! < .5 THEN Pause! = .5   'Force Pause to be within limits
         IF Pause! > 4 THEN Pause! = 4
         PRINT
         WHILE INKEY$ <> "": WEND          'Dump keyboard buffer
         Quit = 0                'Any key sets Quit to exit
         DO UNTIL Quit    'Run in AUTO mode till key is pressed
```

```
            IF INKEY$ <> "" THEN Quit = -1        'Any key aborts
            Cntr = Cntr + 1
            IF Cntr > Npatterns - 1 THEN Cntr = 1  'Wrap around
            Current& = Pattern&(Cntr)
            PauseTime! = TIMER + Pause!
            GOSUB DRIVE
            DO UNTIL TIMER > PauseTime! OR Quit    'Pause between patterns
               IF INKEY$ <> "" THEN Quit = -1     'Any key aborts
               IF TIMER > TimeOut! AND hot THEN 'Wait hot cycle, then go to PWM sustain cycle
                  SEND "x"                 'Turn off hot cycle mode
                  hot = 0                   'Prevent doing so again
               END IF
            LOOP
         LOOP
         PRINT " End auto cycle": PRINT
         GOSUB DRIVE
      CASE Teach$                 'Ctrl-T teaches currently displayed pattern
         PRINT "Teaching pattern #"; Cntr
         Pattern&(Cntr) = Current&     'Store current bit pattern
         Cntr = Cntr + 1
         IF Cntr > 500 THEN Cntr = MaxPatterns  'Not too many
         IF Cntr > Npatterns THEN    'Appending new patterns vs. reteaching
            Npatterns = Cntr
            Pattern&(Cntr) = Pattern&(Cntr - 1) 'Clone last pattern
         END IF
         Current& = Pattern&(Cntr)
         PRINT : GOSUB DRIVE
      CASE Help$, F1$, "?"        'Help keys
         GOSUB HELPSCREEN         'Print cheat sheet
         PRINT : GOSUB DRIVE
      CASE New$                   'Ctrl-N, enter new keyboard layout
         PRINT "Train new keyboard layout"
         PRINT "  Are you sure?  (Y/N)";
         DO: Y$ = UCASE$(INKEY$)         'Confirm new layout request
         LOOP WHILE Y$ = ""
         PRINT " "; Y$
         IF Y$ = "Y" THEN
            FOR Char = 0 TO 255          'Clear out old lookup table
               Layout&(Char) = 0
            NEXT Char
            PRINT
            FOR Element& = 0 TO 29    'Train new key <--> element pattern
               LOCATE CSRLIN - 1: PRINT "       Element #"; Element&
               DO: Key$ = UCASE$(INKEY$)        'Get a key
               LOOP WHILE Key$ = ""
               Layout&(ASC(Key$)) = 2& ^ Element&    'Binary bit pattern
            NEXT Element&
            OPEN "KEYBITS.ARY" FOR OUTPUT AS #1  'Save new array
               FOR Char = 0 TO 255
                  PRINT #1, Layout&(Char)
               NEXT Char
            CLOSE #1
         END IF
         PRINT : GOSUB DRIVE
      CASE Load$
         PRINT
         SHELL "DIR *.PAT"            'Get directory of all *.PATtern files
         INPUT "Load file name"; File$
         IF INSTR(File$, ".") THEN File$ = LEFT$(File$, INSTR(File$, ".") - 1) 'Strip extension if
any
         File$ = File$ + " PAT"
         OPEN File$ FOR INPUT AS #3
            INPUT #3, Npatterns
            FOR I = 1 TO Npatterns
               INPUT #3, Pattern&(I)
            NEXT I
         CLOSE #3
         Cntr = 1
         Current& = Pattern&(Cntr)
         PRINT : GOSUB DRIVE
      CASE Save$
```

```
         PRINT
         INPUT "Save file name"; File$
         IF INSTR(File$, ".") THEN File$ = LEFT$(File$, INSTR(File$, ".") - 1) 'Strip extension if
any
         File$ = File$ + ".PAT"
         OPEN File$ FOR OUTPUT AS #3
            PRINT #3, Npatterns
            FOR I = 1 TO Npatterns
            PRINT #3, Pattern&(I)
            NEXT I
         CLOSE #3
         PRINT : GOSUB DRIVE
      CASE ELSE
         Element& = Layout&(ASC(Key$))   'Get the bit number from lookup table
         IF Element& <> 0 THEN
            IF Current& AND Element& THEN            'IF element is ON
               Current& = Current& AND NOT Element&   'Then turn OFF
            ELSE
               Current& = Current& OR Element&        'Else turn ON
            END IF
            GOSUB DRIVE
         END IF
      END SELECT

   GOTO MAINLOOP        'Repeat the keyboard fetch


DRIVE:     '***********  SEND NEW BIT PATTERN  ************
   LOCATE CSRLIN - 1: PRINT "Pattern #"; Cntr;
   FOR Byte = 0 TO 24 STEP 8
      FOR Bit = 0 TO 7
         IF BitMask&(Bit + Byte) AND Current& THEN
            PRINT "1";
         ELSE
            PRINT "0";
         END IF
      NEXT Bit
      PRINT " ";
   NEXT Byte

   NewOnes& = (Current& XOR LastOnes&) AND Current&  'Figure out which elements just got turned on
   IF hot THEN SEND "x"     'Go to cool cycle prematurely if key was hit
      'Turn on (or leave on) the elements in the current pattern
      '  ... and set the newly turned on ones to full duty cycle (hot cycle)
   SEND "0" + HEX$(Current&) + ". o 0" + HEX$(NewOnes&) + ". h"
      '  (Leading zero avouds misinterpretation of "B." and "D.")


   hot = -1                         'We are running at 100% duty cycle -- hot cycle
   TimeOut! = TIMER + HotTime!   'Set timer to switch to low duty cycle after 'HotCycle' seconds,
resolution = .01 sec

   LastOnes& = Current&


   PRINT SPACE$(10)

RETURN

SUB SEND (OutStr$)

   DO WHILE NOT EOF(2)
      A$ = INPUT$(1, #2)
'      PRINT A$;
   LOOP

   FOR Char = 1 TO LEN(OutStr$)              'One character at a time
      PRINT #2, MID$(OutStr$, Char, 1);      'Send one character
      DO WHILE LOC(2) = 0                    'Wait for echo
      LOOP
```

```
      DO WHILE NOT EOF(2)
         A$ = INPUT$(1, #2)
'         PRINT A$;
      LOOP
   NEXT Char
   PRINT #2,                          'Terminate transmission with a RETURN

   TimeOut! = TIMER + .1              'Wait for OK prompt
   DO UNTIL TIMER > TimeOut!
   LOOP
   'INPUT ; Dmy$

   DO WHILE NOT EOF(2)               'Read in OK prompt
      A$ = INPUT$(1, #2)
'      IF A$ <> CHR$(10) AND A$ <> CHR$(13) THEN PRINT A$;  'Don't print CR's
   LOOP


END SUB
```

```
'*************************************************
'*            TACTILE DEVICE DRIVER              *
'*              A SIMPLE EXAMPLE                 *
'*              For Tini Alloy Co.               *
'*             TQ Automation, 7/89               *
'*************************************************

'This program loops through a list of patterns and sends them to the device

HotTime! = .4        'Number of seconds to hold newly activated elements at full
duty cycle (effective minimum .3)
DwellTime! = 1       'Number of seconds to delay AFTER HOT CYCLE until the next
pattern
Npatterns = 6        'Number of patterns to be cycled
DIM Pattern&(Npatterns)     'Dimention the pattern array

FOR I = 1 TO Npatterns      'Read in the pattern array from data statements below
   READ Pattern$(I)
NEXT I

INIT:        '******  INITIALIZE THE DRIVER BOARD  ********
   PRINT "...Initializing the driver board"
      'NOTE: Change the following statement for COM1 or COM2 as needed
   OPEN "COM2: 9600,N,8,1,CD0,CS0,DS0,OP1000,RS, " FOR RANDOM AS #2
   OutStr$ = "COLD"                    'Do a cold reset
   GOSUB SEND                          'Send the string on the serial port
   OutStr$ = "HEX"                     'Switch to hexadecimal number base
   GOSUB SEND
   OutStr$ = "8000 Period !"           'Set cycle period (lower = higher
                              frequency)
   GOSUB SEND
   OutStr$ = "6000 Ontime !"           'Set duty cycle (lower = shorter duty
                                 cycle)
   GOSUB SEND
                                       '     ... duty cycle = Ontime / Period
   OutStr$ = "58on"                    'Turn on the PWM timer
   GOSUB SEND

START:
   Counter = Counter + 1                       'Increment the pattern counter
   IF Counter > Npatterns THEN Counter = 1     'Wrap around to first pattern
   Current& = Pattern&(Counter)                'Get the pattern you want to send
   NewOnes& = (Current& XOR LastOnes&) AND Current&  'Figure out which elements
                                                     just got turned on
      'This is where we format the data to send to the driver board.
     'We stack up things to store in registers when the time comes.
     'HotReg and OnReg are variable names on the driver board.
     'All extra '0's and '.'s are needed to send double precision numbers.
     'These events happen in reverse order as indicated by the numbers:
   OutStr$ = "0. HotReg "              '3 - turn off phase B on all elements
   OutStr$ = OutStr$ + "0" + HEX$(Current&) + ". OnReg "  '2 - turn on phase A o
             all currently selected elements
   OutStr$ = OutStr$ + "0" + HEX$(NewOnes&) + ". HotReg " '1 - turn on phase B o
                elements that were previously off
```

APPENDIX E

```
(     TACTILE DEVICE APPLICATION PROGRAM     )
(              For Tini Alloy C.             )
(           A. Soghikian 6/10/89             )


HEX
RAMVAR FlipFlop      ( ON / OFF togle flag )
RAMVAR HotReg 2 RAMptr +! 0. HotReg 2! ( Stores bits that remain ON )
RAMVAR OnReg 2 RAMptr +! 0. OnReg 2!   ( Stores bits that are ON )
RAMVAR Period  6700 Period !        ( PWM timer interrrupt )
RAMVAR Ontime  3000 Ontime !       ( PWM on-Period, initial value, chan
A )
RAMVAR IntSpace 1E RAMptr +!         ( Create interrupt stack, same as
ALLOT )
IntSpace 1E + CONSTANT IntStack    ( Top of interrupt stack )

: 58Int  ( -- )        ( Hi level OC1 int handler for PWM on SPI device
)
    -1 FlipFlop @ -              ( Toggle flip flop )
    DUP FlipFlop !
    IF                        ( If -1 then ON cycle, else OFF )
       OnReg 2@ 4OUT              ( Turn specified bits ON )
       Ontime @ 1000 +            ( Get ON cycle time )
       TOC1 +!           ( Set int timer to change to OFF cycle )
    ELSE
       HotReg 2@ 4OUT              ( Turn bits off, except Hot ones )
       Period @
       Ontime @ - 1000 +          ( Calc OFF cycle time )
       TOC1 +!           ( Set int timer to change to ON cycle )
    THEN
    80 TFLG1 C!                   ( Reset interrupt )
    ;

CODE lo58Int ( -- )   ( Interrupt handler )
    CC C, ' 58Int CFA ,         ( LDD # CFA of high level int handler )

    18 C, CE C, IntStack ,       ( LDY # IntStack )
    BD C, ATO4 ,                 ( JSR ATO4, call Forth word in D )
    3B C,                        ( RTI )
END-CODE

: 58Inst ( -- )            ( Install PWM interrupt routine at OC1 )
   [ ' lo58Int @ ] LITERAL       ( Install interrupt routine at OC1 )
   TOC1VEC 1+ E!                 (    ... JMP location )
   7E TOC1VEC EC!                (    ... JMP op-code )
   ;

: 58on ( -- )   ( Enable PWM interrupts )
    58Inst              ( Make sure PWM is installed )
    0 FlipFlop !         ( Preset flip flop )
    Period @ 2 / Ontime !
```

```
    0 OC1M C!              ( OC1 event no effect )
    0 OC1D C!              ( OC1 event no effect )
    C0 TCTL1 C!            ( OC1 disconnected )
    80 TMSK1 C!            ( Enable OC1 interrupt )
    intOn                  ( Allow interrupt processing )
    ;

: 58off ( -- )    ( Disable PWM )
    0 TMSK1 C!             ( Disable OC1 interrupt )
    ;

: x ( -- )    ( Turns off hot cycle )
   0. HotReg 2!
   ;

: h ( d -- )   ( Turns on specified bits for hot cycle )
   HotReg 2!
   ;

: o ( d -- )   ( Turns on specified bits for PWM on cycle )
   OnReg 2!
   ;


RAMVAR n 2 RAMptr +! 0. n 2!                  ( Which bits to turn ON )
RAMVAR HotTime   10 HotTime !   ( Duration of full-ON period )
RAMVAR Scale     16 Scale !

: buzz ( d -- )    ( Short, high current pulse, then PWM )
    0 4OUT           ( Clear all bits in 5818 )
    2DUP n 2!        ( Saves which wires you want ON )
    OnReg 2!         ( Turns on specified bits )
    0. HotReg 2!     ( Warm mode )
    CR ." 5818 Wire Driver Adjustment.   <CR> to quit "
    CR ."   Period:   Duty:  Surge: "
    CR
    58Inst
    58on
    DECIMAL              ( Display numbers in decimal )
    BEGIN                ( Loop start )
      HiA/D@                          ( Read all channels of A/D )
      ADR4 C@ DUP 100 * DUP 8 U.R Period !  ( Get Period of PWM )
      ADR3 C@ * DUP Ontime !                ( Get duty cycle )
      64 UM*                          ( Convert to percent )
      Period @ UM/MOD 8 U.R DROP      (       "            )
      ADR2 C@ 40 * DUP HotTime !      ( Get duration of Hot cycle )
      Scale @ / 8 U.R                 ( Change to miliseconds )
      Pant @ msec                     ( Wait for terminal )
      D EMIT                          ( No line feed )
      ?TERMINAL IF                        ( Parse keyboard if any key
depressed )
        KEY
        DUP 31 = IF          ( Press 1 to turn ON hot cycle )
           n 2@ OnReg 2!        ( Turn ON bit )
```

```
            n 2@ HotReg 2!          ( Keep bit ON full cycle )
            HotTime @ msec          ( Hold the bit high of HotTime counts )

            0. HotReg 2!            ( Enable partial cycle PWM )
         THEN
         DUP 32 = IF             ( Press 2 to turn ON PWM only )
            n 2@ OnReg 2!           ( Turn ON bit )
            0. HotReg 2!            ( Enable partial cycle PWM )
         THEN
         DUP 33 = IF             ( Press 2 to turn ON PWM only )
            0. OnReg 2!             ( Turn OFF bit )
            0. HotReg 2!            ( Enable partial cycle PWM )
         THEN
         D = IF                  ( Press CR to quit )
            58off                   ( Kill PWM )
            0. 4OUT                 ( Clear all bits )
            HEX                     ( Back to Hex format )
            CR ABORT                ( Quit <<<------  EXIT LOOP -------- )
         THEN
       THEN
     0 UNTIL          ( Endless loop )
     ;

INIT-SPI


         (      AUTOSTART OVERHEAD      )

HEX
: auto ( -- )    ( Autostart routine executed on RESET, COLD, or POWERUP
)
    ( main                        Execute main program )
    3F7F 6 80 CMOVE             ( Restore user area in zero page )
    C100 DP !                   ( Put dictionary in RAM )
    INIT-SPI                    ( Initialize the SPI bus )
    58Inst                      ( Install the PWM interrupt vector )
    0 OUT 0 OUT 0 OUT 0 OUT     ( Clear all element driver bits )
    0. HotReg 2!                ( Reset Hot Register )
    0. OnReg 2!                 ( Reset On Register )
    HEX
    ;

: ?auto ( -- ) ( Autostart setup help )
    CR
    CR ."  Auto start setup, after LOADing: "
    CR ."  HEX 2000 AUTOSTART auto      'Autostart patch vector "
    CR ."  A44A 2000 !                  'Put autostart pattern in ROM "
    CR
    ;

C100 DP !                       ( Move dictionary into RAM )
C000 4A !                       ( Put FENCE up to protect ROM from FORGET )

6 3F7F 80 CMOVE                 ( Copy zero page user area to top of ROM )
```


*U.S.Government Printing Office:1990—548-018/20214